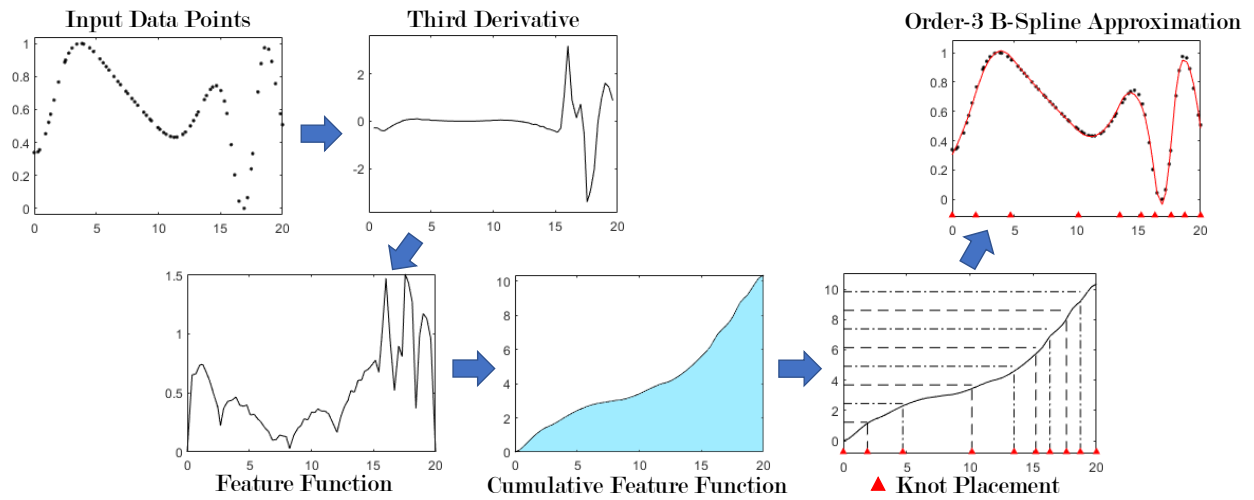


Graphical Abstract

Fast Automatic Knot Placement Method for Accurate B-spline Curve Fitting

Raine Yeh, Youssef S. G. Nashed, Tom Peterka, Xavier Tricoche



Highlights

Fast Automatic Knot Placement Method for Accurate B-spline Curve Fitting

Raine Yeh, Youssef S. G. Nashed, Tom Peterka, Xavier Tricoche

- We automatically compute a set of knots that enable low error approximation for smooth datasets.
- Our algorithm is fast, run-time scaling linearly with the input data size.
- We can give a good estimate of the number of knots needed for a given error threshold.

Fast Automatic Knot Placement Method for Accurate B-spline Curve Fitting[★]

Raine Yeh^{a,*}, Youssef S. G. Nashed^b, Tom Peterka^c and Xavier Tricoche^a

^aDepartment of Computer Science, Purdue University, West Lafayette, IN 47907, USA

^bStats Perform, Chicago, IL 60601, USA

^cMathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, USA

ARTICLE INFO

Keywords:

B-spline curve fitting, knot placement, knot optimization, smooth approximation

ABSTRACT

The choice of knot vector has immense influence on the resulting accuracy of a B-spline approximation of a curve. However, despite the significance of this problem and the various solutions that were proposed in the literature, optimizing the number and placement of knots remains a difficult task. This paper presents a novel method for the approximation of a curve by a B-spline of arbitrary order, which automatically determines a knot vector that achieves high approximation quality. At the core of our approach is a feature function that characterizes the amount and spatial distribution of geometric details in the input curve by estimating its derivatives. Knots are then selected in such a way as to evenly distribute the feature contents across their intervals. A comparison to the state of the art for a wide variety of curves shows that our method is faster and achieves more accurate reconstruction results, while typically reducing the number of necessary knots.

1. Introduction

Curve fitting using B-splines is a fundamental problem in many applications such as computer-aided design (CAD), geometric modeling, and reverse engineering [9, 29]. High-accuracy fitting is also being explored in data analysis and compression for large scale simulations [21]. The problem of B-spline curve fitting involves finding a B-spline curve that minimizes the least squares error between a sequence of input data points and the fitted spline [22, 19]. In this type of fitting problem, variables include the order of the B-spline, the number of knots, the knot locations, and the control point values.

Often, a uniform knot vector with a predetermined number of knots is used. Fixing the knot vector and optimizing only the control points reduces the B-spline fitting problem to a linear least squares problem. However, using uniform knots may fail to capture details of the input dataset. In contrast, solving for the knot vector in addition to the control points can improve the fitting result dramatically [10], leading to the problem of knot optimization.

Knot optimization consists of finding the placement of as few knots as possible for a B-spline curve that fits some desired approximation error criterion. This is a challenging problem for two reasons. First, the unknown number and locations of knots result in a large and nonlinear optimization problem, which is computationally difficult. Second, analytic expressions for optimal knot locations, or even for general characteristics of optimal knot distributions for a desired error criterion, are not easy to derive [10].

In this work, we use the derivatives of the input data


to calculate a feature function that captures the amount and distribution of detail in the data, where higher feature values indicate that a higher knot density is needed to capture the detail and reach the desired approximation error tolerance. Using the cumulative distribution function (CDF) of the feature function, interior knots are distributed, such that higher feature values result in more knots. This approach affords the fitted B-spline more flexibility in those locations, thereby reducing the approximation error. Figure 1 shows an overview of our knot placement method for an approximation using an order-3 B-spline. Given an input dataset (Fig. 1a) and its third derivative (Fig. 1b), our method calculates a CDF of the feature function (Fig. 1c) using the third derivative. Knot locations, indicated by the black triangles, are determined by a set amount of variation of the feature function (Fig. 1d). The approximation is solved using the resulting knot vector, successfully capturing the detail of the input data points (Fig. 1e).

Our approach works for 1-dimensional input data and parametric curves in 2 or higher dimensions. It is fast and produces high accuracy approximation for a wide range of input data that are smooth and sampled densely enough for high-order derivatives to be estimated from the data. The order of the derivatives depends on the order of the B-spline used for the approximation. The knot placement method works for a B-spline approximation of any order, with the resulting approximation error close to a user-supplied target error.

The remainder of the paper starts with a presentation of related work in Section 2. Section 3 briefly reviews technical details of B-spline approximation. We present our method in Section 4, and compare it with prior work in Section 5. Finally, conclusions are drawn, and future work is discussed in Section 6.

[★]This research project is funded by the U.S. Department of Energy, under Contract DE-AC02-06CH11357.

*Corresponding author

 yeh10@purdue.edu (R. Yeh)

ORCID(s):

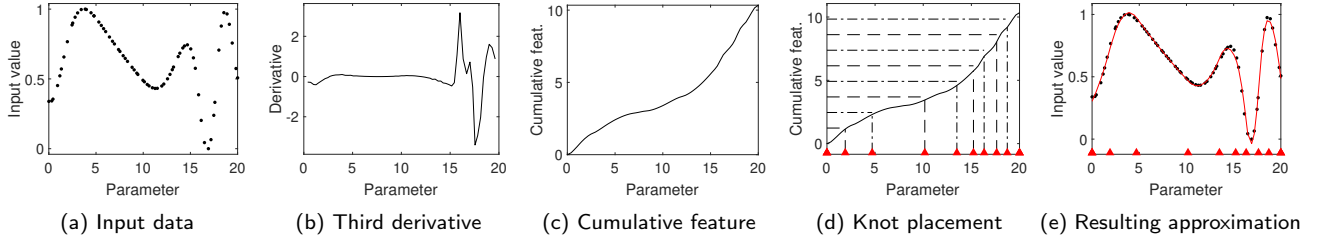


Figure 1: Overview of our method using order-3 B-spline to approximate an input data.

2. Related Work

Knot optimization for B-spline fitting is a well studied topic. Many approaches can be found in the literature. Pieg and Tiller [22] proposed their new knot placement method (NKTP) that places knots with averages of representative parameters for groups of input points. This leads to a stable system of equations and a uniform distribution of knots along the parameter domain. Liang et al. [14] describe an iterative knot insertion (IKI) method that starts with the fewest knots possible, finds knot segments whose approximation error is higher than a given error tolerance, and adds a new knot in the middle of these segments. Dung and Tjahjowidodo [5] propose a fast method for knot placement that locally searches for the largest knot spans under an error threshold using binary search. This method is fast and uses few knots, but often produces a discontinuous approximation. Knot removal techniques start with a set of dense knots, and iteratively remove knots while maintaining the approximation tolerance [18, 17].

Jupp [10] and Loach and Wathen [15] describe local optimization techniques that transform the constrained optimization problem into an unconstrained problem; then a local gradient-based or Gauss-Newton method is employed for minimization. Global optimization can avoid the drawbacks of local methods, but it is computationally more expensive [2].

Kang et al. [11] and Looock et al. [16] treat knot placement as a convex optimization problem, where the norm of jump of the $(p - 1)^{\text{th}}$ derivative of a p -order B-spline is minimized. These optimization methods compute the number and positions of the knots simultaneously and achieve low approximation error with few knots, but are typically computationally more expensive than other approaches.

A machine learning approach using support vector machines for knot placement is described by Laube et al. [12]. The performance of their approach depends strongly on the training dataset, limiting the applicability to a different dataset than the training dataset. Yuan et al. [33] select knots by extracting optimal subsets from a multiresolution B-spline basis using regression analysis.

There also exists a body of work using genetic algorithms for knot vector optimization [32, 25, 28, 34, 27], meta-heuristics such as the firefly algorithm [7], and elitist clonal selections [8]. Such methods are typically computationally expensive, and often produce globally suboptimal solutions.

Another body of literature proposes heuristic methods that use specific properties of the input dataset to guide knot placement. Curvature, in particular, is a widely-used criterion in heuristic methods. Park and Lee [20] select knots using dominant points, which are points of interest of the input dataset. Initial dominant points are set to points of high curvature; then, additional dominant points are added in segments of high approximation error using the input data curvature. Li et al. [13] place initial knots at zero crossings of the curvature, then iteratively add new knots to balance the integral of curvature of the new knot segments. Aguilar et al. [1] use curvature peaks as initial knots, then iteratively add new knots or adjust existing knots to reduce curvature deviation. Razdan [24] uses curvature and arc length of the input dataset to select points of interest, and construct the B-spline approximation by interpolating those points. This interpolation, however, only used the points of interest and does not take into account the points that were not chosen, which can lead to overall higher error.

Derivatives are also used by some heuristic methods for knot placement. Corresponding techniques approximate the derivatives of the input data using a piecewise low-order polynomial function; the connecting points of the piecewise polynomial are then used as knot locations. Tjahjowidodo et al. [26] find knots for a cubic B-spline approximation by using piecewise linear approximation of the second derivative of the input data. Conti et al. [4] find a smooth fit of noisy input data, calculate the third derivative of the smooth fit, and find the piecewise constant approximation of the derivative.

Finally, other heuristic methods use wavelet decomposition [30].

Though our work also follows a heuristic approach using the derivatives of the input dataset, it produces more accurate approximation in a shorter time compared to the existing heuristic approaches, as we will show in Section 5.

3. Preliminary

We review the basics of B-spline in this section, and refer interested readers to Farin [6] for more in-depth material.

A B-spline of order p is a piecewise polynomial function of order p (degree $p - 1$) with n control points, and is defined

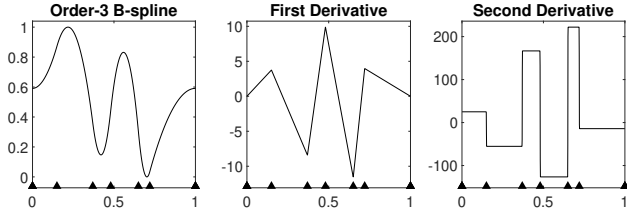


Figure 2: Points sampled from a quadratic (order-3) B-spline and the first two derivatives. The knots, represented by black triangles, mark the derivative's discontinuities.

by

$$\mathbf{C}(u) = \sum_{i=1}^n N_{i,p}(u) \mathbf{P}_i, \quad u \in [t_p, t_{n+1}], \quad (1)$$

where $\mathbf{C}(u)$ is the B-spline curve at parameter location u , \mathbf{P}_i are the control points, and $N_{i,p}$ are the p^{th} order B-spline basis functions defined over the knot vector $\mathbf{T} = \{t_1, t_2, \dots, t_{n+p}\}$. We define our B-spline knot vector with clamping, so the first and last p knots are the same, that is, $t_1 = t_2 = \dots = t_p$ and $t_{n+1} = t_{n+2} = \dots = t_{n+p}$. The B-spline basis function is defined recursively as

$$N_{i,1}(u) = \begin{cases} 1 & \text{if } t_i \leq u < t_{i+1} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

$$N_{i,p}(u) = \frac{u - t_i}{t_{i+p} - t_i} N_{i,p-1}(u) + \frac{t_{i+p+1} - u}{t_{i+p+1} - t_{i+1}} N_{i+1,p-1}(u).$$

The $(p-1)^{\text{th}}$ derivative of an order- p B-spline is piecewise constant. Discontinuities in its derivative coincide with the knot locations of the B-spline. Figure 2 shows an order-3 B-spline and its first two derivatives, with knots indicated by triangles at the bottom. The first derivative is piecewise linear, and the second derivative is piecewise constant.

We consider a sequence of m input data points $Q = \{q_i : q_i \in \mathbb{R}^d\}_{i=1}^m$ with parameterization $U = \{u_i : u_i \in \mathbb{R}\}_{i=1}^m$, where each u_i is a parameter for q_i , and $u_i < u_{i+1}$. For points in 1D space ($d = 1$), $q_i = y_i$, and for points in 2D space ($d = 2$), $q_i = (x_i, y_i)$. Parameters for 1D data points are typically given. For points in 2D space, we calculate the parameters for each point using their chord length $c_j = \sqrt{(x_j - x_{j-1})^2 + (y_j - y_{j-1})^2}$ for $j > 1$. Then the parameters go from 0 to 1, and are defined as $u_1 = 0$ and $u_i = \sum_{j=2}^i c_j / \sum_{j=2}^m c_j$ for $i = 2, \dots, m$.

The B-spline approximation of a set of input points is found in the least squares sense by minimizing the 2-norm of the approximation error:

$$\arg \min_{\mathbf{P}} \sum_{j=1}^m \|q_i - \mathbf{C}(u_i)\|_2 \quad (3)$$

Given the knot vector \mathbf{T} , the control points \mathbf{P}_i of the B-spline curve \mathbf{C} can be found by solving a linear least squares system.

The accuracy of the approximation can be measured by normalized max error

$$E_{\max} = \frac{1}{Q_{\text{rng}}} \max_i \|q_i - \mathbf{C}(u_i)\|_2 \quad (4)$$

or normalized root mean squared error

$$E_{\text{RMS}} = \frac{1}{Q_{\text{rng}}} \sqrt{\frac{1}{m} \sum_{i=1}^m (q_i - \mathbf{C}(u_i))^2} \quad (5)$$

where Q_{rng} is the range of the data. For 1D input data, $Q_{\text{rng}} = y_{\text{rng}}$, and for 2D input data, Q_{rng} is the maximum length of an edge of the axis-aligned bounding box, $Q_{\text{rng}} = \max(x_{\text{rng}}, y_{\text{rng}})$, with $x_{\text{rng}} = x_{\max} - x_{\min}$, and $y_{\text{rng}} = y_{\max} - y_{\min}$.

4. Methodology

Our work is motivated by the idea that an order- p B-spline has a piecewise constant $(p-1)^{\text{th}}$ derivative, where derivative discontinuities mark the knot locations, as shown in Figure 2. Given a set of points sampled from an order- p B-spline, one can recover the original B-spline knots by locating the discontinuities in the $(p-1)^{\text{th}}$ derivative. In practice, the input data points will not be sampled from a B-spline, and will not, in general, have distinct discontinuities in their derivatives. Nonetheless, we show in the following that the derivative information of the input data can be used to guide the knot placement such that resulting knots align with the properties of the input data, thus yielding a better approximation.

Figure 3a shows an example input dataset and its gradually increasing second derivative. More knots are needed on the right side to reduce the approximation error where second derivative is steeper. A knot placement method that does not take the data complexity into account and allocates knots uniformly (such as the NKTP method [23]) will result in insufficient knots on the right side, and therefore exhibit higher error there (as shown in Figure 3c with the NKTP method).

Using the $(p-1)^{\text{th}}$ derivative directly does not solve this problem either. Figure 3b shows a piecewise constant function approximating the second derivative such that the maximum absolute difference between the derivative and the piecewise constant function is minimized. The knots are derived from the breakpoints of the piecewise constant approximation. Similar ideas were attempted in prior work [4, 26]. The right plot in Figure 3b shows the approximation error, where higher error occurred on the left while most knots gather on the right side where the second derivative is steeper.

In this work, instead of using the $(p-1)^{\text{th}}$ derivative directly, we calculate a feature function (Section 4.2) that better captures the amount of detail present throughout the dataset. We then place knots such that each knot segment has the same integral of the feature function (Section 4.3). Figure 3d shows the approximation of the same dataset using

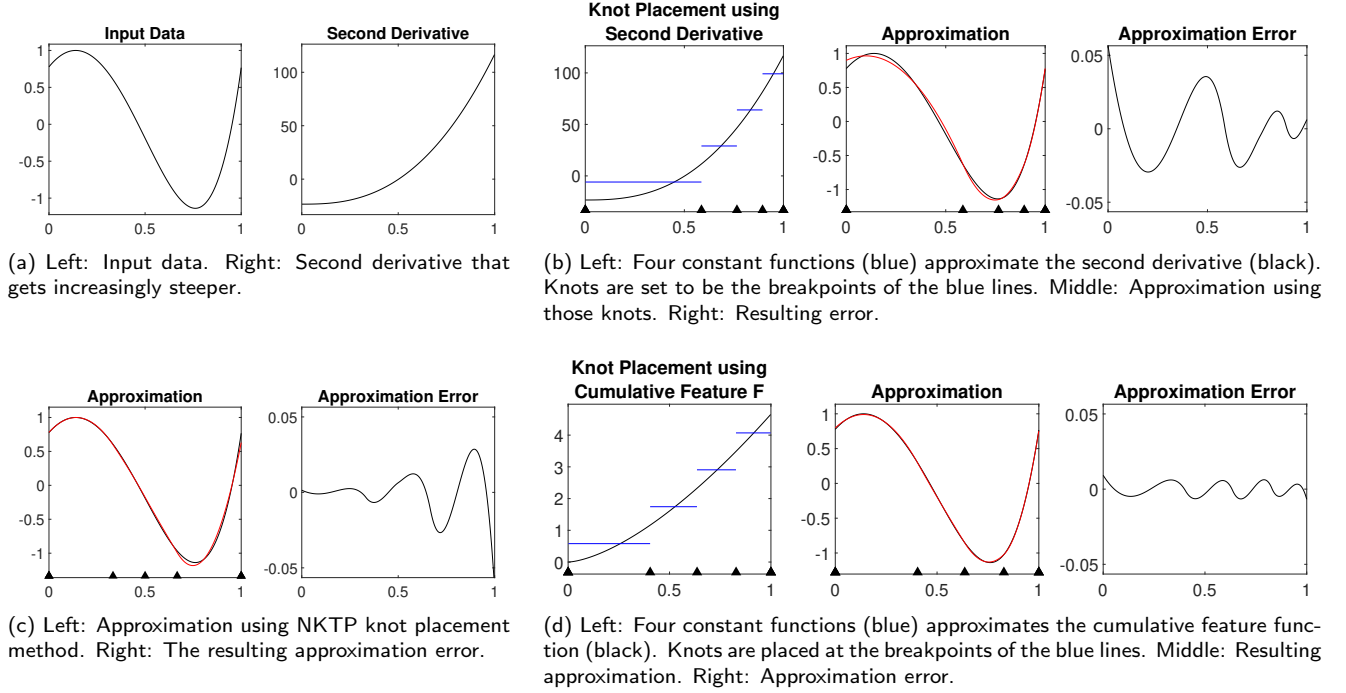


Figure 3: Demonstration of different knot placement methods. The approximated curve is red. The knots used are indicated by black triangles. Each approximation uses the same number of knots, varying only the location of the interior knots.

our method. The resulting approximation exhibits an error that is evenly spread across the domain, producing a higher accuracy approximation using the same number of knots.

Following this idea, our approach for finding the knot vector for B-spline curve fitting is comprised of the following steps:

1. Calculate the derivatives of the input data.
 2. Calculate a feature curve for the input data.
 3. Determine a parameter that decides the number of knots to use.
 4. Adjust the feature curve to avoid a rank deficient system.
 5. Determine the knot vector using the feature curve.
 6. Use least squares minimization to obtain a B-spline approximation of the input data.
- The above steps are explained in greater detail below.

4.1. Derivative Calculation

We use central differences to calculate an approximation of the derivatives of the input points. Central differences is second-order accurate in the parameter spacing. With a given set of m input points $Q = \{q_i : q_i \in \mathbb{R}^d\}_{i=1}^m$ and parameters $U = \{u_i : u_i \in \mathbb{R}, u_i < u_{i+1}\}_{i=1}^m$, we define $Q^{(k)} = \{q_j^{(k)} \in \mathbb{R}^d\}_{j=1}^{m-k}$ to be the set that approximates k^{th} derivatives of the input points at parameters $U^{(k)} = \{u_j^{(k)} \in \mathbb{R}\}_{j=1}^{m-k}$. We let $Q^{(0)} = Q$, $U^{(0)} = U$, then for $k > 0$, we use central

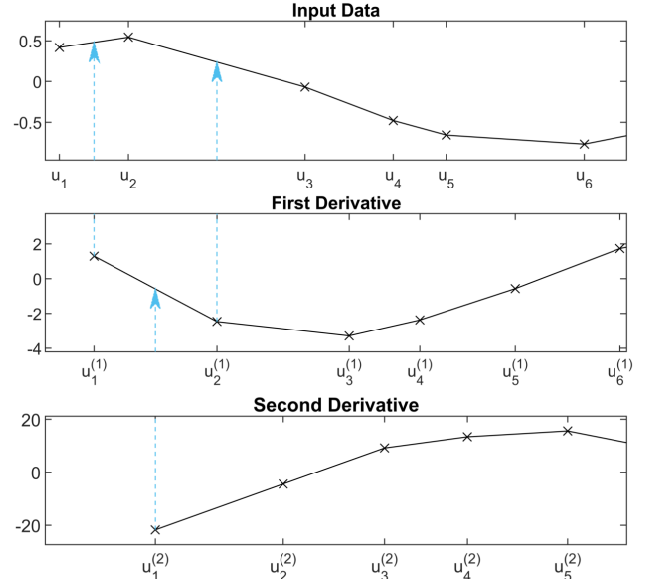


Figure 4: Example of derivative calculation. The parameter location of each derivative value is the midpoint of two points from the previous level, as indicated by the blue dashed arrows.

differences to find $q_j^{(p)}$

$$q_j^{(k+1)} = \frac{q_{j+1}^{(k)} - q_j^{(k)}}{u_{j+1}^{(k)} - u_j^{(k)}} \quad (6)$$

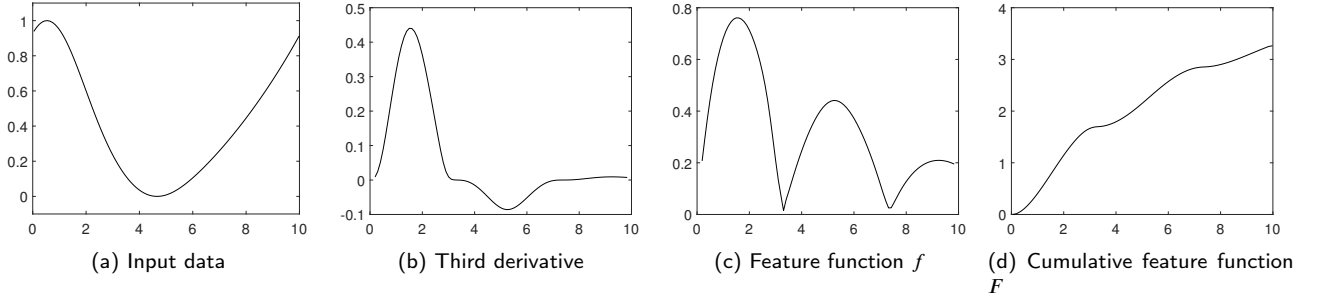


Figure 5: Example of calculating the feature function f and the cumulative feature function F

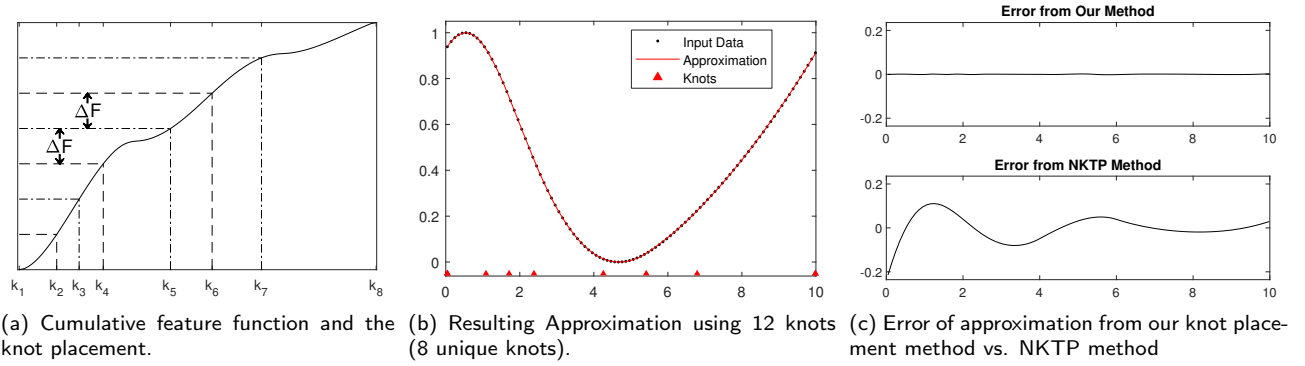


Figure 6: Placement of knots using the cumulative feature function F

with parameter

$$u_j^{(k+1)} = \frac{1}{2} \left(u_j^{(k)} + u_{j+1}^{(k)} \right). \quad (7)$$

Note that each level of derivatives has its own set of parameters, which are midpoints of the parameters of the previous level derivatives.

Figure 4 shows an example of calculating the first and second derivatives of the starting segment of a dataset.

Other methods of derivative calculations can also be applied here. In fact, if the analytical derivatives are known, the proposed method will automatically benefit from the additional accuracy.

Care must be taken when calculating derivatives of noisy data because differentiation in general, and central differences in particular, are known to amplify the noise. However, the present work does not consider the specific issue of fitting noisy data, a problem for which different solutions have been proposed. For instance, Conti et al. [4] fit a smoothing B-spline with dense knots to the noisy input data, and estimate the derivatives using the smoothing B-spline.

4.2. Calculating Feature Function

The feature function $f(u)$ measures the amount of detail in the input data points, and is later used for knot placement in the B-spline approximation step. The feature function is defined using a set of *feature points* $\{f_i\}$, calculated from the p^{th} derivative of the input dataset via a normalization

function Φ , where p is the order of B-spline used to approximate the data.

We define the set of feature points $\{f_i\}$ at parameter locations $\{\bar{u}_i\}$, $0 \leq i \leq m - p + 1$, as

$$(\bar{u}_i, f_i) = \begin{cases} (u_1, 0), & i = 0 \\ (u_i^{(p)}, \Phi(\|q_i^{(p)}\|_2)), & 1 \leq i \leq m - p \\ (u_m, 0), & i = m - p + 1 \end{cases} \quad (8)$$

We take the magnitude of the p^{th} derivatives of the input data points, and use a normalization function Φ to balance the knot distribution and prevent the case in Figure 3b, where too many knots are allocated at steeper derivatives.

Empirically, we find that defining Φ to be the p^{th} root of the p^{th} derivative produces the best error distribution. Thus, the definition of f_i for $i = 1, \dots, m - p$ is

$$f_i = \left(\|q_i^{(p)}\|_2 \right)^{1/p}. \quad (9)$$

Then we define the continuous feature function $f(u)$ to be the piecewise linear interpolant of the set of feature $\{f_i\}$. In other words, for parameter u where $\bar{u}_i \leq u \leq \bar{u}_{i+1}$,

$$f(u) = \frac{u - \bar{u}_{i+1}}{\bar{u}_i - \bar{u}_{i+1}} f_i + \frac{u - \bar{u}_i}{\bar{u}_{i+1} - \bar{u}_i} f_{i+1} \quad (10)$$

for $i = 0, \dots, m - p + 1$, and $f(u) = 0$ for u outside of the range $[\bar{u}_0, \bar{u}_{m-p+1}]$.

The feature function $f(u)$ represents the amount of detail at parameter location u . The higher the value of $f(u)$, the smaller the knot span at the parameter location u .

Figures 5a to 5c show the feature set calculation process for a sample input. For an order-3 B-spline approximation, the third derivative is calculated in Figure 5b, and the feature function is shown in Figure 5c. The cumulative curve of Figure 5d is described in the next subsection.

4.3. Knot Placement

We now consider the problem of how to distribute the knots using the feature function $f(u)$.

We want to place knots such that the integral of f over each knot span is the same. To do so, we calculate $F(u)$, the cumulative distribution function (CDF) of $f(u)$.

$$F(u) = \int_{-\infty}^u f(v) dv, \quad (11)$$

which is equivalent to F being a linear interpolant of the set $\{F_i\}$ at parameter locations $\{\bar{u}_i\}$, $0 \leq i \leq m - p + 1$, where $F_0 = 0$, and

$$F_i = \sum_{j=1}^i f_{j,\text{trap}}. \quad (12)$$

for $i = 1, \dots, m - p + 1$, and $f_{j,\text{trap}}$ is the finite integral approximation of $f(u)$ between \bar{u}_{j-1} and \bar{u}_j calculated using the trapezoid rule

$$f_{j,\text{trap}} = \frac{1}{2} (f_j + f_{j-1}) (\bar{u}_j - \bar{u}_{j-1}). \quad (13)$$

Since F is a cumulative distribution function of the non-negative function f , F starts from zero and is non-decreasing.

Figure 5d shows the cumulative feature function F calculated from the feature function f shown in Figure 5c.

Next, we define F^{-1} as the inverse of F such that

$$F^{-1}(q) = u \Leftrightarrow F(u) = q. \quad (14)$$

The inverse function F^{-1} is well defined if the values $\{F_i\}$ are monotonically increasing, which may not be the case if there are consecutive zeroes in $\{f_i\}$. For the cases where F contains flat spots, we can add a small positive value η to the integration

$$f_{j,\text{trap}} = \frac{1}{2} (f_j + f_{j-1} + \eta) (\bar{u}_{j-1}^{(p)} - \bar{u}_j^{(p)}) \quad (15)$$

to ensure that $\{F_i\}$ and therefore F is monotonically increasing, and F^{-1} is uniquely defined in the whole domain.

With F^{-1} defined, we can now find the knot locations. With knot clamping, the first and last knots are repeated p times. The knot vector \mathbf{T} contains r unique knots, and is defined as

$$\mathbf{T} = \underbrace{\{k_1, \dots, k_1\}}_p, \underbrace{\{k_2, \dots, k_{r-1}\}}_p, \underbrace{\{k_r, \dots, k_r\}}_p \quad (16)$$

where the range of the knots is the same as the range of the input data parameter, that is, $k_1 = u_1$ and $k_r = u_m$.

We find the locations of the r unique knots $\{k_1, \dots, k_r\}$ with

$$k_i = F^{-1}((i-1)\Delta F) \quad (17)$$

where ΔF is the amount of integrated feature per knot segment. The selection of ΔF determines the number of knots used and the resulting accuracy of the approximation. A smaller ΔF results in greater number of knots with shorter knot spans and a higher approximation accuracy, and conversely for larger ΔF . This knot placement ensures that each knot span has the same amount of increase in F ; i. e., $F(k_{i+1}) - F(k_i) = \Delta F$ for all i .

Figure 6a shows the cumulative feature function F split into equal ΔF steps with horizontal dashed lines. The knot locations are shown on the x-axis. If we know r , the number of unique knots, in advance, we can calculate $\Delta F = F_{\max}/(r-1)$, where F_{\max} is the largest value in F . Otherwise, the selection of ΔF is discussed in Section 4.5.

After the knot vector is acquired, the fitting B-spline can be solved with Eq. (3).

Figure 6b shows the approximation using the acquired knot vector, and Figure 6c shows the resulting approximation error of our method (top) compared with the approximation error using the NKTP method [23] (bottom).

4.4. Limiting Knot Density

There may be cases where the resulting knot vector has smaller knot spans than the input data point spacing. This can occur when some part of the cumulative feature function F increases too quickly, resulting in knots placed too close together with the given ΔF . This could result in a rank-deficient least squares system and an inefficient usage of knots.

We prevent this situation by ensuring that knots are not placed too closely to each other by imposing the condition

$$F_i - F_{i-1} \leq \Delta F \quad (18)$$

for all i . We achieve this by limiting F_i to be

$$F_i = \sum_{j=1}^i \min(\Delta F, f_{j,\text{trap}}). \quad (19)$$

We then proceed to place knots as described in Section 4.3.

Figure 7 shows an input dataset sampled from a cosine function, with sparse samples toward the right. Below the cosine plot are the input point parameters shown as vertical bars. Blue downward triangles are knots placed before the adjustment, and red upward triangles are the knots created from the adjusted F function. The pre- and post-adjustment knots match on the left side of the plot where the density of the input points is high enough for the knot density; but on the right side, the pre-adjusted knots are denser than the input points. The post-adjusted knots account for the input point spacing.

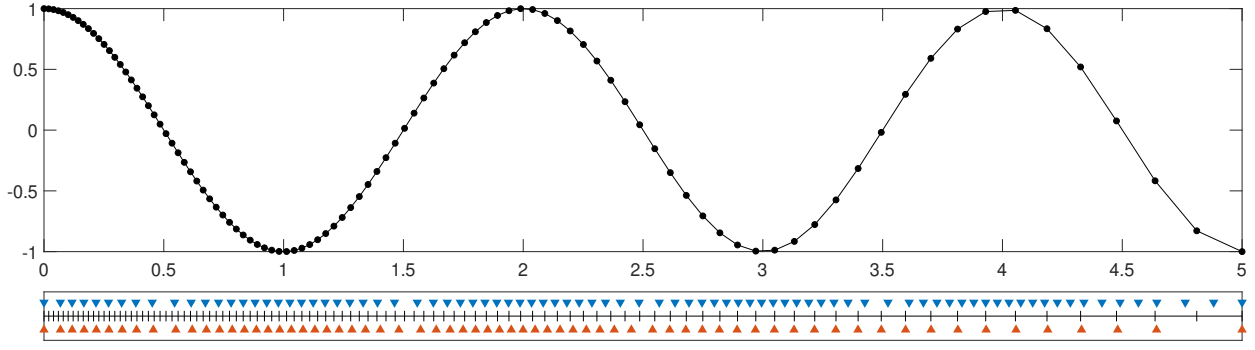


Figure 7: Input data with nonuniform spacing. Blue triangles indicate old unadjusted knots. Red triangles are adjusted knots. Black bars between the triangles indicate input spacing. At the right side the new knots (red) are adjusted to match the spacing of the input points, whereas old knots (blue) are more densely placed, causing rank deficiency in the least squares system.

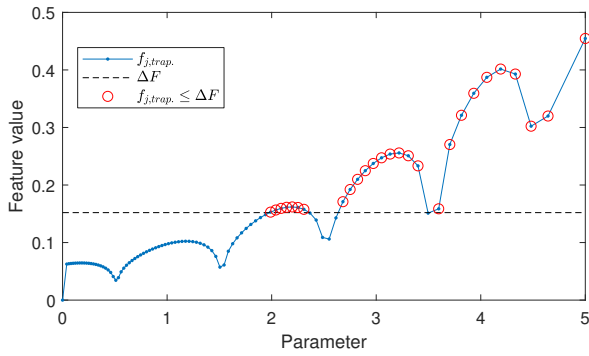


Figure 8: Finite integral approximation $f_{j,trap}$ and the ΔF line that shows the cutoff to prevent a rank-deficient system. All $f_{j,trap}$ above the cutoff are circled.

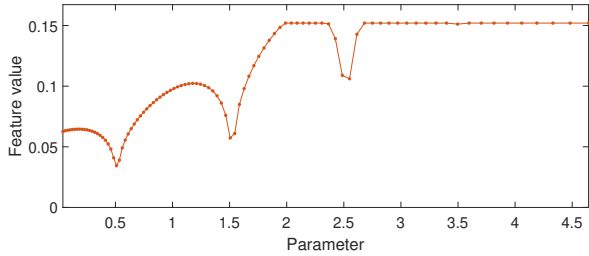


Figure 9: The plot of $\min(f_{j,trap}, \Delta F)$.

Figure 8 shows the finite integral approximation $\{f_{j,trap}\}$ of the input data's feature function and the chosen ΔF . The $f_{j,trap}$ points over the ΔF value are circled. Figure 9 shows the finite integral approximation $\{f_{j,trap}\}$ limited by the ΔF value, and Figure 10 shows the adjusted cumulative feature function F compared with the pre-adjusted F .

4.5. Determining Number of Knots

During the placement of knots, ΔF determines the number of knots used. However, usually users do not know how to set ΔF or the number of knots for a desired approximation quality.

In our knot placement method, we approximate the num-

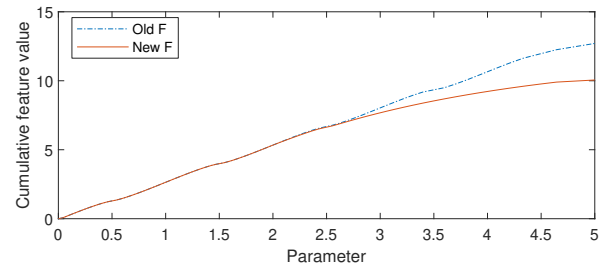


Figure 10: The old and new cumulative feature functions G diverge where $f_{j,trap}$ exceeds ΔF .

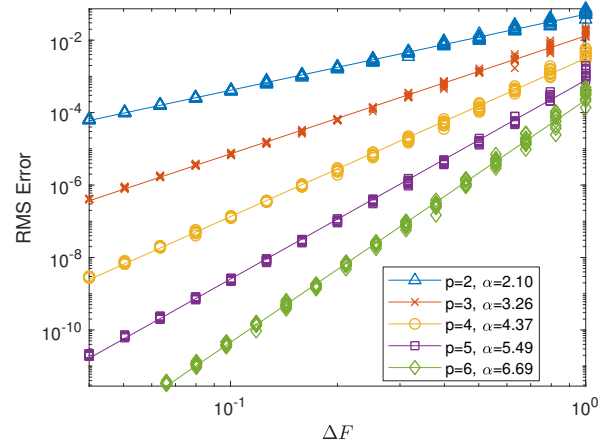


Figure 11: Plot of ΔF value against RMS error for 10 datasets with 5 different B-spline orders. Lines are fitted via linear regression in the log-log space. The slope of each line is α .

ber of knots needed for a desired target error by using regression to find the relationship between the approximation error and ΔF for a specific order- p B-spline.

We tested 10 randomly generated 1D order-9 nonuniform rational B-spline (NURBS) datasets, with random control points, weights, and number of knots ranging from 10 to 40 with randomized knot locations. The range of number of knots results in various feature functions F for each dataset. We use a range of ΔF and B-spline degrees to approximate

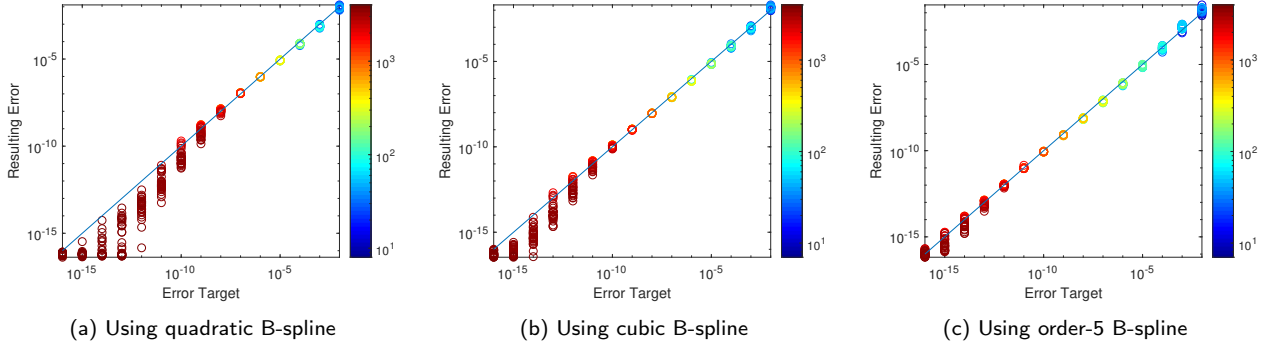


Figure 12: Target error vs. resulting error using three different B-spline orders for 32 randomly generated datasets. Number of knots used for each approximation is color-coded.

these datasets. Figure 11 plots, in log-log scale, the resulting RMS error of these approximations against the ΔF value used, with a different color for each degree of B-spline used. Exponential convergence can be observed from the plot.

For each order of B-spline, we fitted the data points pertaining to the order in log-log space with the line

$$y = \alpha x + \beta \quad (20)$$

where y is the log of the RMS error, x is the log of ΔF ; α is the slope of the line, and β is the y-intercept of the line for the order- p B-spline. We solve for the unknowns α and β using linear least squares to minimize the fitting error in the log-log space for each B-spline order. Then given a desired B-spline order p and the desired target approximation RMS error e , we can approximate ΔF with $\Delta F = 10^{(\log(e) - \beta)/\alpha}$.

We evaluate the line model with 32 randomly generated NURBS datasets, different from the 10 datasets used in the regression. The 32 datasets are of order 7 to 10, contain 4000 points, with number of knots in the range of 10 to 60, with random knot locations, control points, and weights. We check the correlation between the input desired tolerance and the actual deviation error for order-3, 4, and 5 B-splines in the correlation plots in Figure 12. The $x=y$ line is drawn for reference to see how close the resulting error is to the target error. Each approximation point is colored by the number of knots used. Points below the line indicate a resulting error lower than the target error, whereas points above indicate a resulting error higher than the target error.

The resulting error matches the target error for the most part, except toward the lower error, where a faster convergence is observed for quadratic B-spline, and similarly but less noticeably for the other two higher order B-splines. The drop of resulting error is caused by the number of knots approaching the number of input points in each data sets, resulting in interpolation of the input datasets. The lowest errors are around 10^{-16} , limited by machine precision.

The integral of the feature function f (in other words, F_{\max} , the maximum value in F) can be thought of as a measure of how complicated the input data are. The more complicated the input data are, the more knots should be used to achieve the same error threshold.

Table 1

Table of all Methods

Ref.	Label	Descriptions
	Our	Instant; derivative-guided
[22]	NKTP	Instant; parameter-only
[14]	IKI	Iterative
[26]	LinFit	Locally iterative; derivative-guided
[4]	ConFit	Locally iterative; derivative-guided
[1]	AdpCrv	Locally iterative; curvature-guided
[20]	DOM	Iterative; curvature-guided

This is a heuristic linear regression to guide the choice of the number of knots given a desired error tolerance. It uses a small number of datasets with similar characteristics as the target data, and does not guarantee the resulting approximation error to be under the tolerance. The linear regression model can be improved upon with more sophisticated machine learning approaches that generalize to a wider variety of datasets.

5. Experimental Results

The performance of our knot placement algorithm was tested against six prior works. We implemented each of the other methods in MATLAB.

Table 1 lists the algorithms, their labels used in the comparison plots, and the main idea of each method. “Instant” refers to placing knots without any iterative process. “Iterative” indicates that a repeated adjustment of the knots is done to improve the approximation. “Locally iterative” indicates that, at each iteration, a small local system is solved, whereas (globally) iterative methods solve the fitting equation (Eq. (3)) at each iteration. “Parameter-only” refers to methods that only use the parameter information to place knots. These methods do not take into account features of the input data. “Derivative-guided” methods use the derivative information to place knots, whereas “curvature-guided” methods make use of the curvature information. “Our” refers to the method in this paper. Tjahjowidodo et al. [26] use piecewise linear polynomial to fit the second derivative, hence

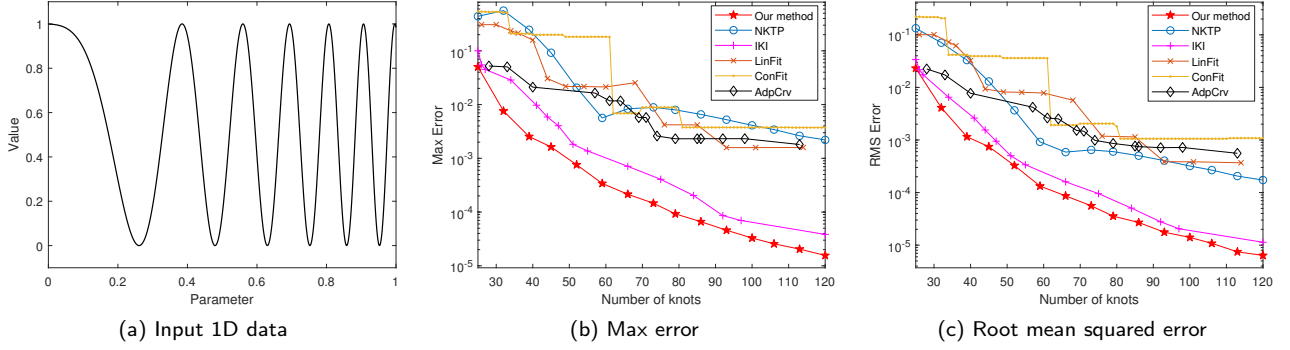


Figure 13: Methods comparison for 801 1D input points uniformly sampled from a cosine with increasing frequency.

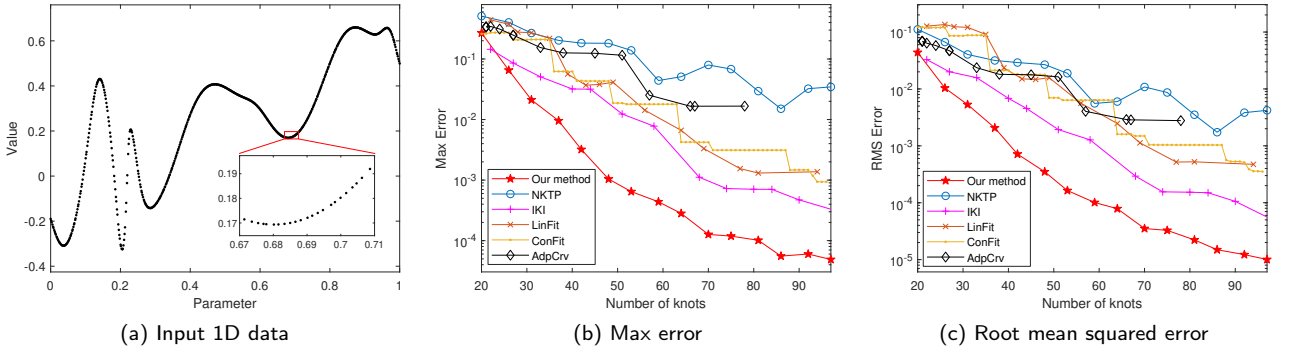


Figure 14: Methods comparison for 501 1D input points sampled nonuniformly from a randomly generated NURBS curve.

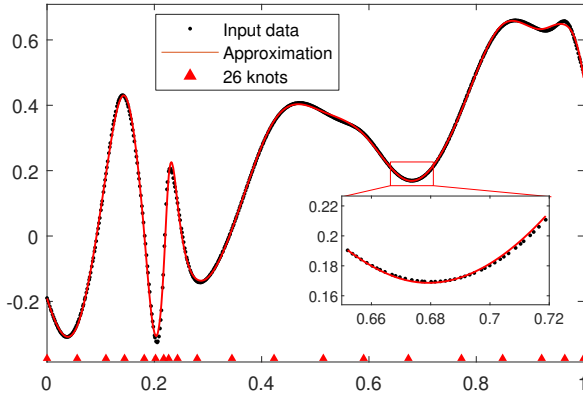


Figure 15: Approximating a NURBS curve using our knot placement method with 26 knots.

their method is dubbed *LinFit*. Similarly, Conti et al. [4] use piecewise constants to fit the third derivative, and we refer to their method as *ConFit*. The method by Aguilar et al. [1] is adaptive curvature guided (*AdpCrv*), and the method by Park and Lee [20] uses dominant points (*DOM*).

The IKI method [14] refines from a starting knot vector. To prevent the starting knots from influencing the result, our implementation of the IKI method starts with knots only at the end points.

The discrete derivatives discussed in Section 4.1 were used for our method for all datasets. Our input data are

free of noise generally, but datasets with varying parameter spacing would result in noise in the derivative computation. Therefore, for some of the other methods that are more sensitive to the smoothness of the derivative and curvature calculation, we calculate the derivatives by fitting a cubic smoothing spline over the data. The fitted spline is then used to calculate the approximate derivatives and curvature of the data. We use MATLAB `csaps` function for the cubic smoothing spline. For *AdpCrv* and *DOM* methods, datasets with varying spacing (Figure 14a and 19a) and the simulation dataset (Figure 16a) used the smoothing weight of 10^{-5} . For the *ConFit* method, the smoothing weight used is 10^{-8} for all 1D datasets.

Some of the methods only work with cubic B-splines (order-4); thus, we use cubic B-splines in our experiments to match these methods. However, note that our knot placement method works for B-splines of any order.

5.1. Approximation Error

In each example case, the input points are plotted first (a), followed by the maximum error (b) and the root mean squared error (c), plotted against the number of knots used. Each method was run 15 times with different numbers of knots, and the resulting approximations were evaluated. The two exceptions are *DOM* and *ConFit* methods. Those methods add one knot at each iteration until a target number of knots is reached. Therefore, the plots for those methods show the progression of error as each knot is added.

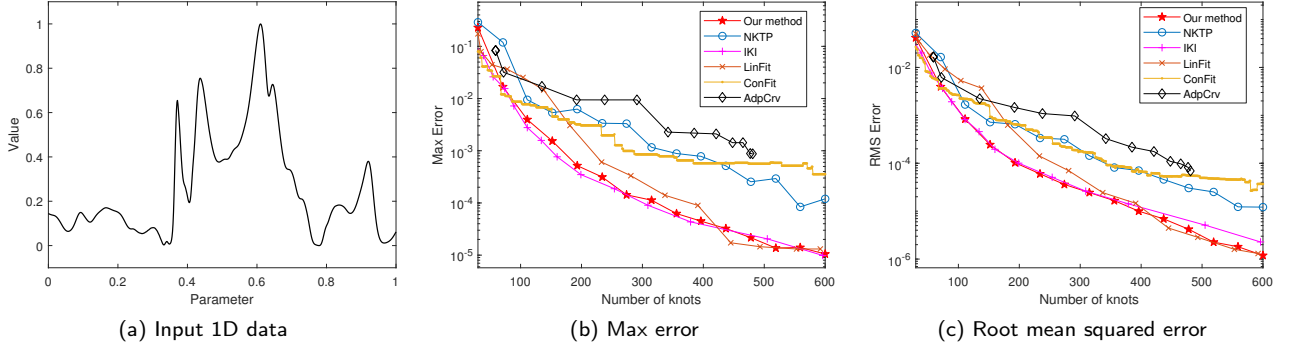


Figure 16: Methods comparison for 704 1D input points taken from a slice of 3D simulation data.

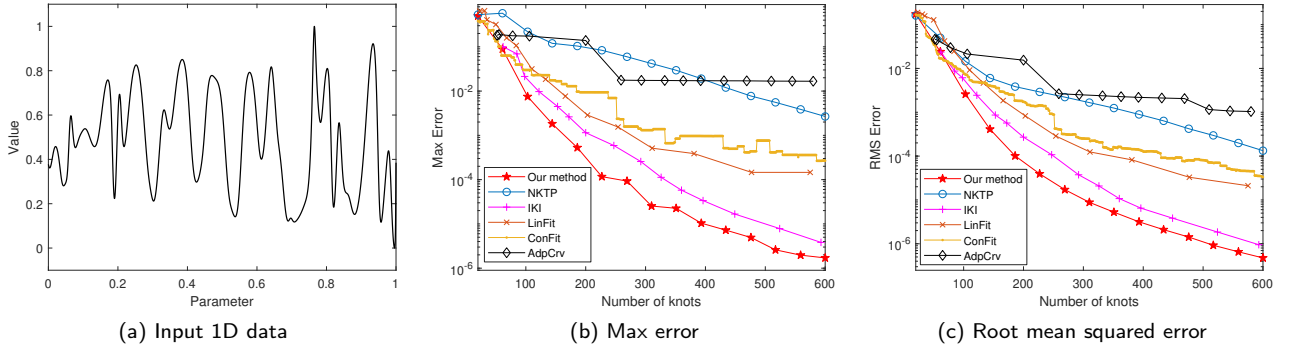


Figure 17: Methods comparison for a 1D randomly generated NURBS with 4000 points.

The test cases are selected to include a range of fitting complexity, with varying amount of detail. Of the eight datasets, the first four are 1D signals, and the other four are parametric curves. Test datasets are comprised of a combination of synthetic and actual scientific data. Some of the synthetic datasets are generated using high-order NURBS with randomly distributed control points, knots, and weights, such that the data cannot be exactly represented using the approximating B-splines.

5.1.1. 1D datasets

Figures 13 to 17 compare the approximation results for the four 1D datasets, using all methods except the DOM method, as DOM is described only for parametric curves.

The first dataset shown in Figure 13a corresponds to 801 points sampled from a cosine wave with increasing frequency, resulting in progressively higher frequency oscillation toward the right side of the data. More knots need to be allocated toward the right to accurately capture the data. As can be seen from the maximum and RMS error plots (Figure 13b and 13c), our method achieves the lowest error for all tested number of knots, followed by the IKI method. The other four methods have higher error for the tested number of knots. The NKTP method does not allocate enough knots toward the right hand side to lower the error sufficiently. The drop at 54 knots for the NKTP method occurs when the knot density is high enough to fit the shape of the right-most oscillation. On the other hand, the three other heuristic methods,

LinFit, ConFit, and AdpCrv methods, place too many knots toward the right side, which quickly increases the number of knots without reducing the approximation error on the left side. Sharp drops in error for the ConFit method occur when knots are added toward the right side.

The second dataset shown in Figure 14a corresponds to 501 points sampled from a randomly generated NURBS curve of order 6 with 30 knots. The control points, knot locations, and weights are randomly generated. The points are sampled with randomly varying point spacing, as shown in the zoom-in plot. This dataset contains a smooth curve with a sharp dip around parameter value 0.2. Figure 15 shows the approximation result using our method and the knot placement with 26 knots. Figures 14b and 14c show the approximation error of all the methods with different numbers of knots. Our method achieves the lowest approximation error for all numbers of knots. The iterative IKI method reaches the second lowest error in the range of the number of knots used, followed by the ConFit and LinFit methods. The NKTP and AdpCrv methods have the highest error for most knot numbers. Since the sampling of the input dataset is uniformly randomized, and not proportional to the amount of variation in the data, the NKTP method does not allocate more knots near the sharp dip at 0.2 parameter.

The third dataset shown in Figure 16a contains 704 points and is a 1D slice of a 3D dataset, measuring the magnitude of the velocity of a turbulent combustion simulation [3]. This dataset contains sharper features compared with previ-

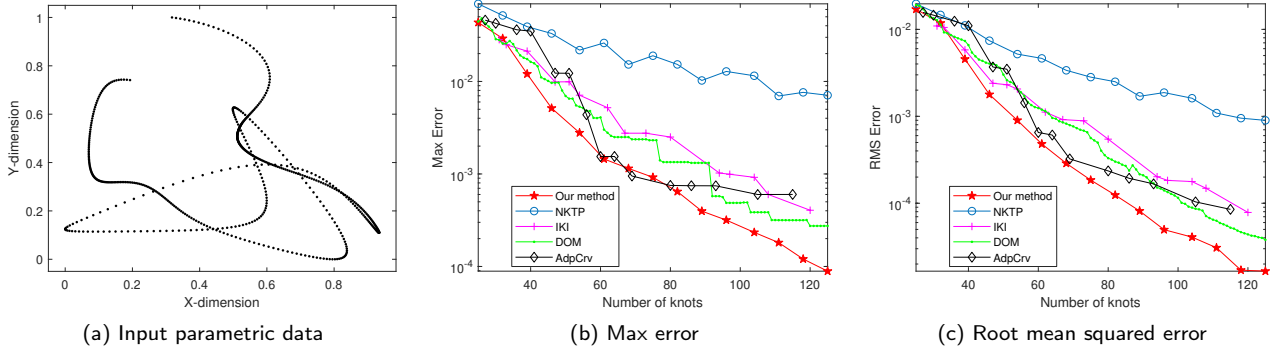


Figure 18: Methods comparison for 601 parametric points sampled from a random NURBS curve.

ous datasets. Figures 16b and 16c show the approximation error for all the methods. For this dataset, our method and the IKI method achieve similar error for the same number of knots. Because of the higher frequency variation present in this dataset, the derivative is noisier compared with the previous datasets. The IKI method tends to perform better with less smooth datasets thanks to its localized refinement step. LinFit also achieves one of the lowest approximation errors at higher knot count, as it captures the sharper turns. ConFit achieves a low error with fewer knots, but the decrease in error slows as more knots are added to locations with highly varying third derivatives but already low error. AdpCrv does not converge as quickly as the other methods due to the more complex nature of the dataset, since more evenly spread knots are needed to prevent an excessive concentration of knots at high curvature locations. The NKTP method lowers the error at a steady rate with more knots added because of the relatively even detail distribution of the dataset across the parameter space.

The fourth dataset shown in Figure 17a is a large randomly generated NURBS dataset with 4000 points and 80 random knots, control points, and weights. The randomized nature and the large number of complex regions represent complicated and unpredictable datasets found in real-world applications. Figures 17b and 17c show the approximation error for all the methods. Overall, our method achieves the lowest approximation error across all number of knots, followed by the IKI method. The AdpCrv and NKTP methods result in the highest RMS and maximum error.

5.1.2. Parametric datasets

The datasets shown in Figures 18-21 are curves in 2 dimensions, parameterized by arc length. Methods LinFit and ConFit are described for only 1D datasets, and thus are not used to evaluate these parametric curves.

The dataset shown in Figure 18a is an order-7 NURBS curve generated using random control points and weights, and 32 random knots. 601 points are uniformly sampled in the parameter space. Figures 18b and 18c show the approximation error for this dataset across all the methods. Our method achieves the lowest error for both maximum and RMS errors, followed by DOM and AdpCrv methods.

The dataset shown in Figure 19a is sampled from the parametric equations $x(u) = u(\cos(2u) + 0.5)$ and $y(u) = u \sin(u)$. 401 points are sampled along the arc length, with higher sampling density where curvature is higher. The sampling spacing contains a small amount of randomized variation, as shown in the zoomed-in plot. Figures 19b and 19c show the approximation error for this dataset for all the methods. Our method achieves the lowest RMS error, but has higher maximum error in a few cases compared to the DOM method. Method AdpCrv has high error for a low number of knots, but achieves comparatively lower error with 50 or more knots. Even though higher sampling density at high curvature benefits the NKTP method, it produces the highest error in most cases.

The data of Figure 20a is a butterfly contour taken from [31]. 600 points are uniformly sampled in parameter space. It is a more challenging dataset than the cases considered so far, with sharper curves and corners. Our method achieves the lowest error for all numbers of knots, followed by the DOM and AdpCrv methods. The heuristic approach of the DOM method hones in on the sharper corners for knot placement, effectively reducing the approximation error. The IKI method has higher error with the same number of knots because the insertion method always splits at the middle of the segment, which is less effective for this dataset due to the localized high curvature. The NKTP method achieves the highest approximation error among all the methods.

Figure 21a shows the last dataset, a parametric curve with 4000 points sampled from a NURBS curve generated using 70 random knots, control points, and weights. This dataset is larger and more complex than the other parametric datasets. Method DOM achieves lower maximum error for low number of knots, but for higher number of knots, our method reaches lower maximum error. For RMS error, our method maintains the lowest error compared with all other methods. The IKI method reduces both maximum and RMS errors consistently as more knots are added, but it does so at a lower rate than DOM and our method. Method AdpCrv does not reduce either error measure beyond around 300 knots. The NKTP method has the highest error compared with all other methods.

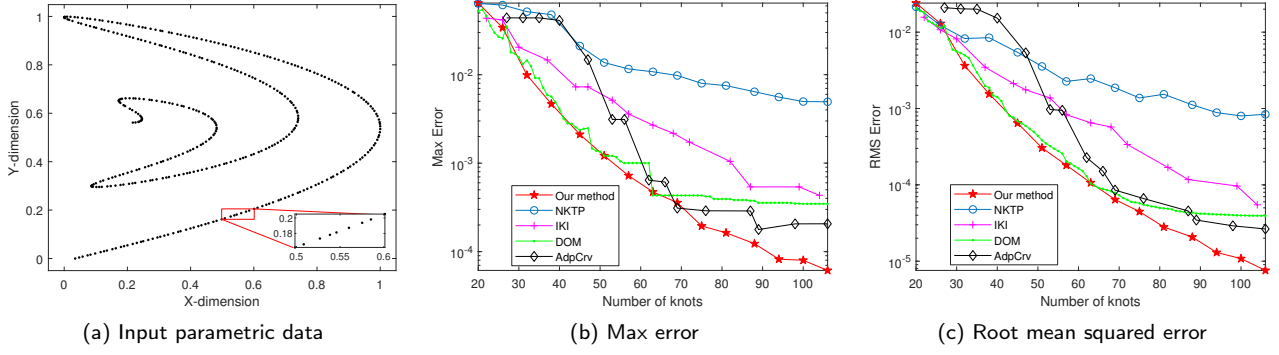


Figure 19: Methods comparison for 401 points nonuniformly sampled from a parametric function.

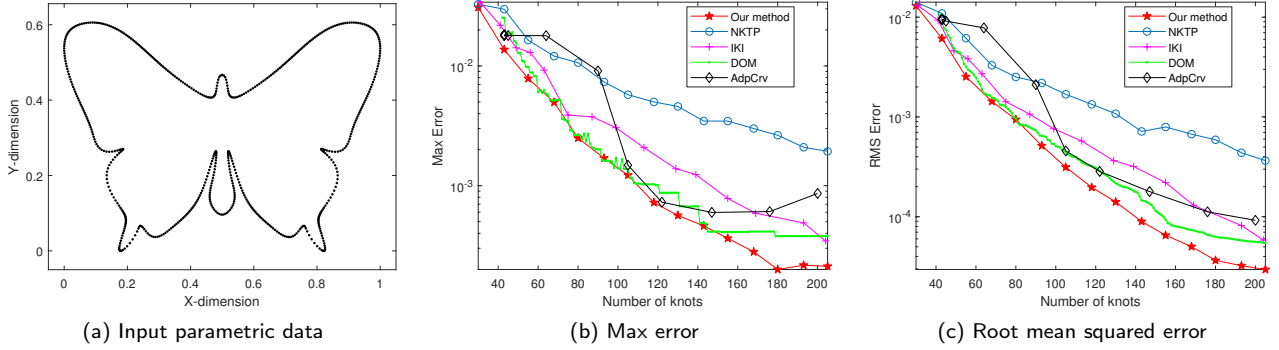


Figure 20: Methods comparison for 600 points sampled from a butterfly contour taken from [31].

5.1.3. Discussion of each method

In general, our knot placement method allocates more knots to segments with higher information content, resulting in reduced approximation error. In most cases, our method achieves lower approximation error with the same number of knots compared with other methods.

The NKTP method has the highest error for most cases since it ignores the features of the input dataset. The IKI method, due to its adaptive refinement strategy, achieves comparatively lower error in the 1D cases. In particular in the simulation dataset, IKI yields results comparable to our method, focusing on the locations where more knots are needed to

reduce error. Due to the chord-length parameterization for curves in 2D, data points may be grouped closely together in a small parameter domain, requiring more refinement steps and knots to improve the approximation in those regions.

Method LinFit is described only for 1D datasets, and is thus not used on the three parametric datasets. It allocates more knots at locations with highly varying second derivatives to capture the detail in the data, but in some cases could result in redundant knots at the locations with highly varying second derivatives. The method's accuracy directly depends on the fitting parameter used, although the relationship between fitting parameter and approximation error is unclear.

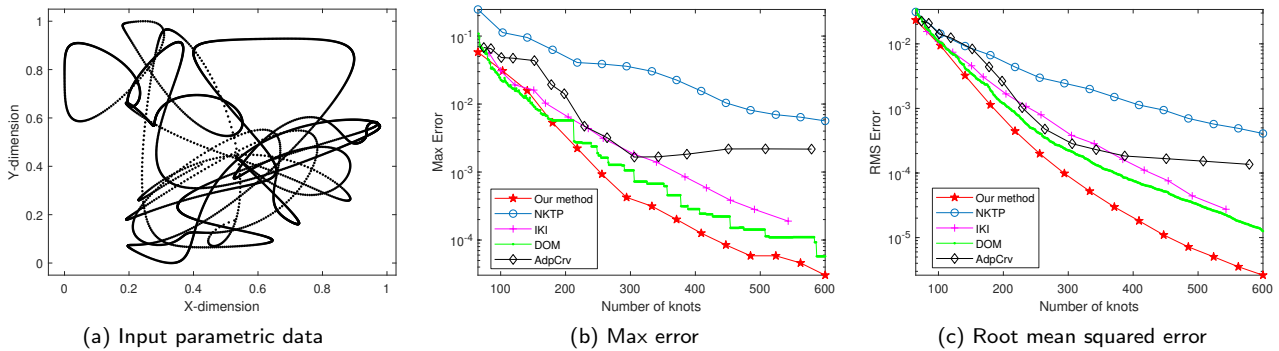


Figure 21: Methods comparison for a randomly generated NURBS parametric curve with 4000 points.

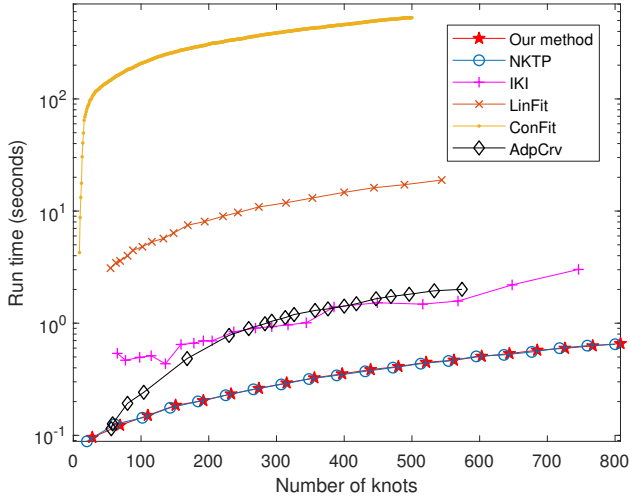


Figure 22: Timing vs. number of knots for 1D data with 4000 points

ConFit is described for 1D datasets only, and is not used on the three parametric datasets. This method may perform better in cases where fewer knots are used to capture the general shape of noisy data or data with few samples, as the cubic smoothing spline used for calculating derivatives filters out the noise in the data. In the present context, however, ConFit does not perform as well as other methods because our test datasets are free of noise, and we aim to approximate to a low error tolerance.

Method AdpCrv, being curvature based, achieves lower error on smooth parametric datasets. For datasets with higher curvature, the method can allocate too many knots in the high curvature regions. The simulation dataset shown in Figure 16a has many curvature peaks, making it a challenging dataset for AdpCrv method.

The DOM method is described only for parametric curves, and is not evaluated with the three 1D datasets. The method's error-driven knot placement reduces the maximum error quickly for low knot count. However, because existing dominant points are not updated, and there must be a minimum of three input data points between any dominant point pair, the maximum error can plateau as more knots are added.

5.2. Timing

We compare the timing of each methods, using a Desktop computer with a 3.3GHz Intel i7-3960X CPU and 32GB RAM. For each parameter used in each method, the timing result reported is the median of ten runs.

Our implementations of the various methods (including our own) are not optimized, thus the resulting timing information is meant to show relative performance for comparison only.

For the algorithms using 1D data, we use the randomly generated dataset shown in Figure 17a with 4000 points. The summary of the resulting timing for all methods on 1D data is shown in Figure 22.

Algorithms operating on parametric curves are compared

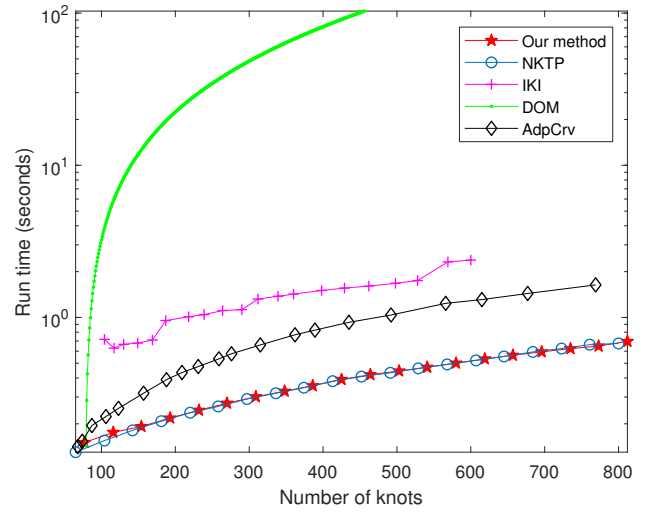


Figure 23: Timing vs. number of knots for parametric data with 4000 points

with the randomly generated dataset shown in Figure 21a, also with 4000 points. Figure 23 shows the timing results for the parametric data.

Since we start with the minimum number of knots for method IKI, it takes many iterations and performs as many least squares solves. We only count the time for iterations that occur after the knot vector becomes non-uniform, for a fairer comparison, which amounts to starting the iterative process with a uniform knot vector.

The fastest run times per knot inserted is obtained with NKTP and our method, as no iterations are needed. These methods are $\mathcal{O}(n)$ where n is the number of input points.

The run times of globally iterative methods (such as DOM and IKI) and locally iterative methods (AdpCrv method and LinFit) are affected by their convergence, which is data dependent. The iterative IKI method solves a linear least squares system at each iteration, and thus requires more time to reach the desired result. The LinFit method fits a piecewise linear function to data points during its local iteration, which we implemented using linear programming. Using a different fitting criterion may improve the speed of the method. Method ConFit performs a neighboring knot adjustment step after each newly inserted knot, which can be time consuming. Method DOM inserts one knot and solves a linear system at each iteration, hence it is the slowest of all considered methods. We implemented the method as described in the paper, but the timing may be improved by adding all the knots for all segments above the error threshold in an iteration. At each iteration, the linear system changes by only a few columns, which could also be leveraged by using an iterative solver to reduce the execution time further. The run time of AdpCrv method depends heavily on the number of local adjustment iterations, which is, in turn, data dependent. The method is likely to perform faster with smoother data as fewer adjustment iterations are needed. This can be seen in the comparison of AdpCrv method between 1D and data parameterized by arc length, which result in lower curvature.

This can explain the faster run time in the parametric case than in the 1D case.

6. Conclusion and Discussion

We introduced a novel knot placement optimization method that analyzes high-order derivatives of the input data, and generates a knot placement such that the resulting least squares fit has low error. We demonstrated our method's effectiveness by comparing it to a number of state-of-the-art knot placement methods, and showed that our method can achieve comparable or higher approximation accuracy using fewer knots for a range of 1D and parametric datasets. Our method is also computationally inexpensive, with run time scaling linearly with the dataset size. In our experiments, our method's execution times were on par with the fastest methods we evaluated.

A challenge for our method is noise in the data, due to the need to calculate higher-order derivatives, which tend to amplify the noise. Methods for computing derivatives of noisy data, e.g., based on an intermediate reconstruction with a smoothing B-spline, will need to be assessed in this context to extend this method for data containing noise.

The regression analysis to determine the number of knots for a given error tolerance is a rough estimate for datasets with a certain convergence rate. Further study can be done on analyzing the smoothness of the dataset to determine its potential convergence rate given a selected B-spline order.

Additional avenues for future work include approximating multidimensional and time-dependent data. This may introduce new challenges depending on how the B-spline basis is extended to multiple dimensions.

Acknowledgments

This work is supported by Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357, program manager Laura Biven.

References

- [1] Aguilar, E., Elizalde, H., Cárdenas, D., Probst, O., Marzocca, P., Ramirez-Mendoza, R.A., 2018. An adaptive curvature-guided approach for the knot-placement problem in fitted splines. *Journal of Computing and Information Science in Engineering* 18. doi:10.1115/1.4040981.
- [2] Beliakov, G., 2004. Least squares splines with free knots: global optimization approach. *Applied Mathematics and Computation* 149, 783 – 798. doi:10.1016/S0096-3003(03)00179-6.
- [3] Chen, J.H., Choudhary, A., de Supinski, B., DeVries, M., Hawkes, E.R., Klasky, S., Liao, W.K., Ma, K.L., Mellor-Crummey, J., Podhorszki, N., Sankaran, R., Shende, S., Yoo, C.S., 2009. Terascale direct numerical simulations of turbulent combustion using s3d. *Computational Science & Discovery* 2, 015001. doi:10.1088/1749-4699/2/1/015001.
- [4] Conti, C., Morandi, R., Rabut, C., Sestini, A., 2001. Cubic spline data reduction choosing the knots from a third derivative criterion. *Numerical Algorithms* 28, 45–61. doi:10.1023/A:1014022210828.
- [5] Dung, V.T., Tjahjowidodo, T., 2017. A direct method to solve optimal knots of B-spline curves: An application for non-uniform B-spline curves fitting. *PLOS ONE* 12, 1–24. doi:10.1371/journal.pone.0173857.
- [6] Farin, G., 2002. *Curves and Surfaces for CAGD: A Practical Guide*. 5th ed., Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- [7] Galvez, A., Iglesias, A., 2013. Firefly algorithm for explicit b-spline curve fitting to data points. *Mathematical Problems in Engineering* 2013. doi:10.1155/2013/528215.
- [8] Gálvez, A., Iglesias, A., Avila, A., Otero, C., Arias, R., Manchado, C., 2015. Elitist clonal selection algorithm for optimal choice of free knots in b-spline data fitting. *Applied Soft Computing* 26, 90 – 106. doi:10.1016/j.asoc.2014.09.030.
- [9] Hoschek, J., Lasser, D., 1993. *Fundamentals of Computer Aided Geometric Design*. A. K. Peters, Ltd., Natick, MA, USA.
- [10] Jupp, D., 1978. Approximation to data by splines with free knots. *SIAM Journal on Numerical Analysis* 15, 328–343. doi:10.1137/0715022.
- [11] Kang, H., Chen, F., Li, Y., Deng, J., Yang, Z., 2015. Knot calculation for spline fitting via sparse optimization. *Computer-Aided Design* 58, 179 – 188. doi:10.1016/j.cad.2014.08.022.
- [12] Laube, P., Franz, M.O., Umlauf, G., 2018. Learnt knot placement in b-spline curve approximation using support vector machines. *Computer Aided Geometric Design* 62, 104 – 116. doi:10.1016/j.cagd.2018.03.019.
- [13] Li, W., Xu, S., Zhao, G., Goh, L.P., 2005. Adaptive knot placement in b-spline curve approximation. *Computer-Aided Design* 37, 791 – 797. doi:10.1016/j.cad.2004.09.008.
- [14] Liang, F., Zhao, J., Ji, S., Fan, C., Zhang, B., 2017. A novel knot selection method for the error-bounded b-spline curve fitting of sampling points in the measuring process. *Measurement Science and Technology* 28, 065015. doi:10.1088/1361-6501/aa6a05.
- [15] Loach, P.D., Wathen, A.J., 1991. On the Best Least Squares Approximation of Continuous Functions using Linear Splines with Free Knots. *IMA Journal of Numerical Analysis* 11, 393–409. doi:10.1093/imanum/11.3.393.
- [16] Look, W.V., Pipeleers, G., Schutter, J.D., Swevers, J., 2011. A convex optimization approach to curve fitting with b-splines. *IFAC Proceedings Volumes* 44, 2290 – 2295. doi:10.3182/20110828-6-IT-1002.00452.
- [17] Lyche, T., 1988. A data-reduction strategy for splines with applications to the approximation of functions and data. *Ima Journal of Numerical Analysis* 8, 185–208. doi:10.1093/imanum/8.2.185.
- [18] Lyche, T., Mørken, K., 1987. Knot removal for parametric b-spline curves and surfaces. *Computer Aided Geometric Design* 4, 217 – 230. doi:10.1016/0167-8396(87)90013-6.
- [19] Ma, W., Kruth, J., 1995. Parameterization of randomly measured points for least squares fitting of b-spline curves and surfaces. *Computer-Aided Design* 27, 663 – 675. doi:10.1016/0010-4485(94)00018-9.
- [20] Park, H., Lee, J.H., 2007. B-spline curve fitting based on adaptive curve refinement using dominant points. *Computer-Aided Design* 39, 439 – 451. doi:10.1016/j.cad.2006.12.006.
- [21] Peterka, T., Youssef S. G., N., Grindeanu, I., Mahadevan, V.S., Yeh, R., Tricoche, X., 2018. Foundations of multivariate functional approximation for scientific data, in: 2018 IEEE 8th Symposium on Large Data Analysis and Visualization (LDAV), pp. 61–71. doi:10.1109/LDAV.2018.8739195.
- [22] Piegl, L., Tiller, W., 1997. *The NURBS Book* (2nd Ed.). Springer-Verlag, Berlin, Heidelberg.
- [23] Piegl, L., Tiller, W., 2000. Surface approximation to scanned data. *The Visual Computer* 16, 386–395. doi:10.1007/PL00013393.
- [24] Razdan, A., 1999. Knot placement for b-spline curve approximation.
- [25] Sarfraz, M., Raza, S.A., 2001. Capturing outline of fonts using genetic algorithm and splines, in: *Proceedings Fifth International Conference on Information Visualisation*, pp. 738–743. doi:10.1109/IV.2001.942138.
- [26] Tjahjowidodo, T., Dung, V., Han, M., 2015. A fast non-uniform knots placement method for b-spline fitting, in: 2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM), pp. 1490–1495. doi:10.1109/AIM.2015.7222752.
- [27] Tongur, V., Ülker, E., 2016. B-spline curve knot estimation by us-

- ing niched pareto genetic algorithm (npga), in: Lavangnananda, K., Phon-Amnuaisuk, S., Engchuan, W., Chan, J.H. (Eds.), *Intelligent and Evolutionary Systems*, Springer International Publishing, Cham. pp. 305–316.
- [28] Valenzuela, O., Delgado-Marquez, B., Pasadas, M., 2013. Evolutionary computation for optimal knots allocation in smoothing splines. *Applied Mathematical Modelling* 37, 5851 – 5863. doi:10.1016/j.apm.2012.11.002.
- [29] Várady, T., Martin, R.R., Cox, J., 1997. Reverse engineering of geometric models — an introduction. *Computer-Aided Design* 29, 255 – 268. doi:10.1016/S0010-4485(96)00054-1.
- [30] Xuming He, Lixin Shen, Zuowei Shen, 2001. A data-adaptive knot selection scheme for fitting splines. *IEEE Signal Processing Letters* 8, 137–139. doi:10.1109/97.917695.
- [31] Yau, H.T., Lin, M.T., Tsai, M.S., 2006. Real-time nurbs interpolation using fpga for high speed motion control. *Computer-Aided Design* 38, 1123 – 1133. doi:10.1016/j.cad.2006.06.005.
- [32] Yoshimoto, F., Moriyama, M., Harada, T., 1999. Automatic knot placement by a genetic algorithm for data fitting with a spline, in: *Proceedings Shape Modeling International '99. International Conference on Shape Modeling and Applications*, pp. 162–169. doi:10.1109/SMA.1999.749336.
- [33] Yuan, Y., Chen, N., Zhou, S., 2013. Adaptive b-spline knot selection using multi-resolution basis set. *IIE Transactions* 45, 1263–1277. doi:10.1080/0740817X.2012.726758.
- [34] Ülker, E., 2013. B-spline curve approximation using pareto envelope-based selection algorithm – pesa. *International Journal of Computer and Communication Engineering* 2, 60–63. doi:10.7763/IJCC.2013.v2.137.